NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A SOFTWARE IMPLEMENTATION OF AN
INTERACTIVE GRAPHICS SYSTEM FOR THREE DIMENSIONAL
MODELING AND LAYOUT

by

Surasak Mungsing

March 1986

Thesis Advisor: Michael J. Zyda

CURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

REPORT SECURITY CLASSIFICATION	1b RESTRICTIVE MARKINGS
SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT
DECLASSIFICATION/DOWNGRADING SCHEDULE	Approved for public release; distribution is unlimited
PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
NAME OF PERFORMING ORGANIZATION: aval Postgraduate School / 6b OFFICE SYMBOL (If applicable): Code 52	7a NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
ADDRESS (City, State, and ZIP Code): onterey, CA 93943-5000	7b ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000
NAME OF FUNDING/SPONSORING ORGANIZATION / 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

TITLE (Include Security Classification)

SOFTWARE IMPLEMENTATION OF AN INTERACTIVE GRAPHICS SYSTEM FOR THREE DIMENSION MODELING AND LAYOUT

PERSONAL AUTHOR(S)

ungsing, Surasak

a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
aster's Thesis	FROM _____ TO _____	1986 March	52

SUPPLEMENTARY NOTATION

COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	INTERACTIVE, GRAPHICS, THREE DIMENSIONAL, LAYOUT, MODELING, WALKTHROUGH, PIPING, MENU

ABSTRACT (Continue on reverse if necessary and identify by block number)

his thesis examines interactive techniques for viewing a 3-D building model in a walk-hrough fashion and for placing 3-D piping into a 3-D building model. The focus of re-earch is software implementation using the C programming language and the IRIS Graphics ibrary on the Silicon Graphics Inc. IRIS Turbo 2400 interactive graphics system. The irst part of the research is concerned with drawing, viewing a 3-D building model, and xamining interactive techniques required for building walkthrough mechanism. The second art is concerned with the development of techniques necessary to allow the placement of -D piping into a 3-D building model using 2-D graphics display and a mouse device. The lgorithms and implementation of these techniques are presented.

D STRIBUTION/AVAILABILITY OF ABSTRACT: XX UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION: unclassified
a NAME OF RESPONSIBLE INDIVIDUAL: Prof Michael Zyda	22b TELEPHONE (Include Area Code): 408 646-2305 / 22c OFFICE SYMBOL: 52Zk

D FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

A Software Implementation of an Interactive Graphics System for Three Dimensional Modeling and Layout

by

Surasak Mungsing
Flight Lieutenant, Royal Thai Air Force
B.S.E., University of South Florida, 1975
M.Eng., Chulalongkorn University, Thailand,1984

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POST GRADUATE SCHOOL
March 1986

ABSTRACT

This thesis examines interactive techniques for viewing a 3-D building model in a walkthrough fashion and for placing 3-D piping into a 3-D building model. The focus of research is software implementation using the C programming language and the IRIS Graphics Library on the Silicon Graphics Inc. IRIS Turbo 2400 interactive graphics system. The first part of the research is concerned with drawing, viewing a 3-D building model, and examining interactive techniques required for building walkthrough mechanism. The second part is concerned with the development of techniques necessary to allow the placement of 3-D piping into a 3-D building model using 2-D graphics display and a mouse device. The algorithms and implementation of these techniques are presented.

TABLE OF CONTENTS

I. INTRODUCTION

A. DISCUSSION

Physical models have been used in many layout design applications. A layout design is an arrangement or plan for assembly and/or installation of objects from models. The purpose behind layout design is to reduce or to eliminate problems and difficulties that may be encountered during the construction phase. In the aircraft industry, mock-ups (models of actual size) of complete airplane exteriors and interiors are required for the layout design of hydraulic/pneumatic piping, electrical wiring, control cabling, electronic/avionic equipment installation, etc. In building and factory piping/wiring layout design, scale models are used. The layout design process can involve many changes, requiring modifications of the design and, perhaps, modifications of the mock-ups and scale models. However, reproducing and modifying the design, mock-ups, or scale models is expensive and time consuming.

An interactive graphics display system is an alternative approach for the layout design task. It provides a fast and interactive method for the creation and modification of a layout design. In interactive graphics, an object is composed of a collection of graphics primitives such as three-dimensional lines, points, and polygons. The organization of these primitives defines a representation for a particular object. The image of an object can be created from its mathematical representation and displayed upon a graphics screen. The mapping process can involve a tremendous number of calculations. With today's VLSI technology, the calculations can be performed in hardware and in a pipeline fashion to provide instantaneous results.

A view of a three-dimensional object can be created by the viewing process shown in Figure 1.1. The process transforms a 3-D object into a 2-D projection plane. Conceptually, an object in 3-D world space is clipped against the 3-D view volume, projected onto a 2-D projection plane and finally scaled into a display device's coordinate system. The display can be interactively changed by modifying the position and orientation of objects by means of a set of interactive input devices. Typical devices include keyboards, buttons, valuators (dials), and xy position indicators or locators.

It is possible to interact with a 3-D layout design on a 2-D graphics display. The visual display of a layout model can be put on a screen by the process shown in Figure 1.2. The output pathway starts with an applications program, a piece of software that maintains the mathematical representation of a model, making calls to the graphics package. The graphics package transforms the data passed to it and passes the transformed data to the device driver. The device driver converts the data received into the opcode streams required by a piece of hardware called the display processing unit (DPU). The DPU then converts the opcode streams into a form that can be used by the refresh system that maintains the display on a cathode-ray tube (CRT).

Control values from the interactive devices are passed to the applications program along the input pathway. These input values are used to make a change in the picture from the application program. The new picture is sent to the screen via the output pathway described above. Typical interactive inputs are dragging and picking. Dragging is a technique for dynamically moving an object around with a locator. This technique includes dynamic scaling and rotation of objects to a desired size and orientation. Picking is a technique that allows the designer to pick an object or collection of objects to be operated upon. However, the operations which concern "getting the picture there" (from the applications program to the display surface) and "manipulating the picture" (by way of some movement of the interactive devices such that a picture change is generated) involve intensive mathematical description encoding and matrix transformation calculations. Only workstations with these capabilities in hardware can handle these operations at a satisfactory level of performance.

B. THESIS OBJECTIVE

The objective of this study is to examine and develop interactive techniques for viewing a 3-D building model in a "walkthrough" fashion and for placing 3-D piping into the 3-D building. A high performance graphics workstation is required as a research tool for this effort as it can provide the human user immediate feedback of visual information in response to any physical control manipulations made. In general, any workstation with leading edge capabilities is suitable for this research. The Silicon Graphics Inc. IRIS Turbo 2400 system has been chosen for this effort as it is available for use in the Graphics and Video Laboratory of the Naval Postgraduate School.

The focus of this research is a software implementation using the C programming language and the IRIS Graphics Library. The first part of the research work concerns drawing, viewing a 3-D building model, and examining the interactive techniques required for building "walkthrough". The second part concerns examining interactive techniques necessary to allow the placement of 3-D piping into a 3-D building using a 2-D graphics display and a mouse device.

C. THESIS CONTENTS

The remainder of this study is devoted to the presentation of the IRIS 2400 system, the development of supporting algorithms and the implementation for a building modeling and a 3-D piping layout design. Chapter 2 introduces the IRIS workstation architecture, its significant features, the IRIS graphics library, and the Gaddis menu system. Chapter 3 describes the building walkthrough and walkthrough movement algorithms. Chapter 4 describes the 3-D piping layout algorithm. Chapter 5 describes the program implementation details. Chapter 6 presents the conclusions of the research work.

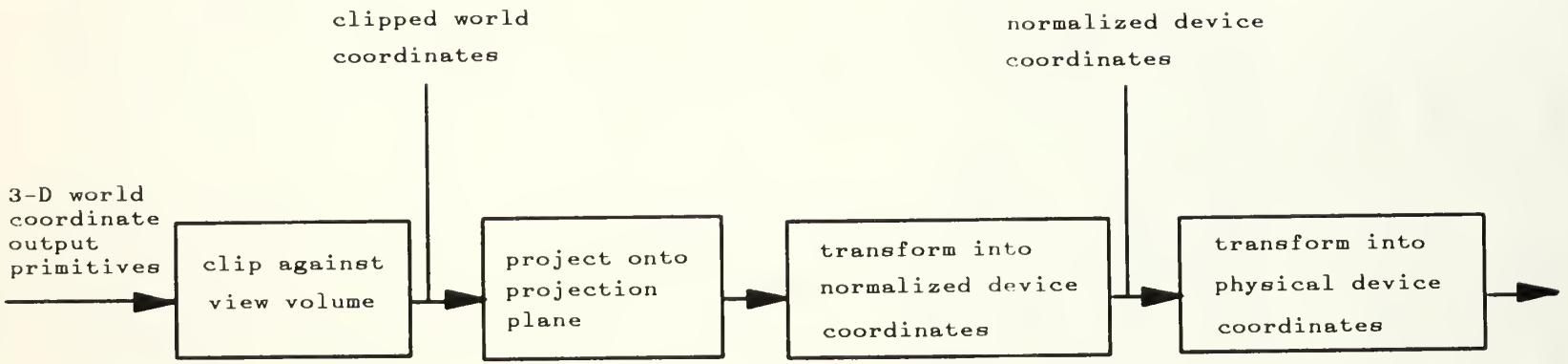


Figure 1.1 Conceptual Model of the 3-D Viewing Process.

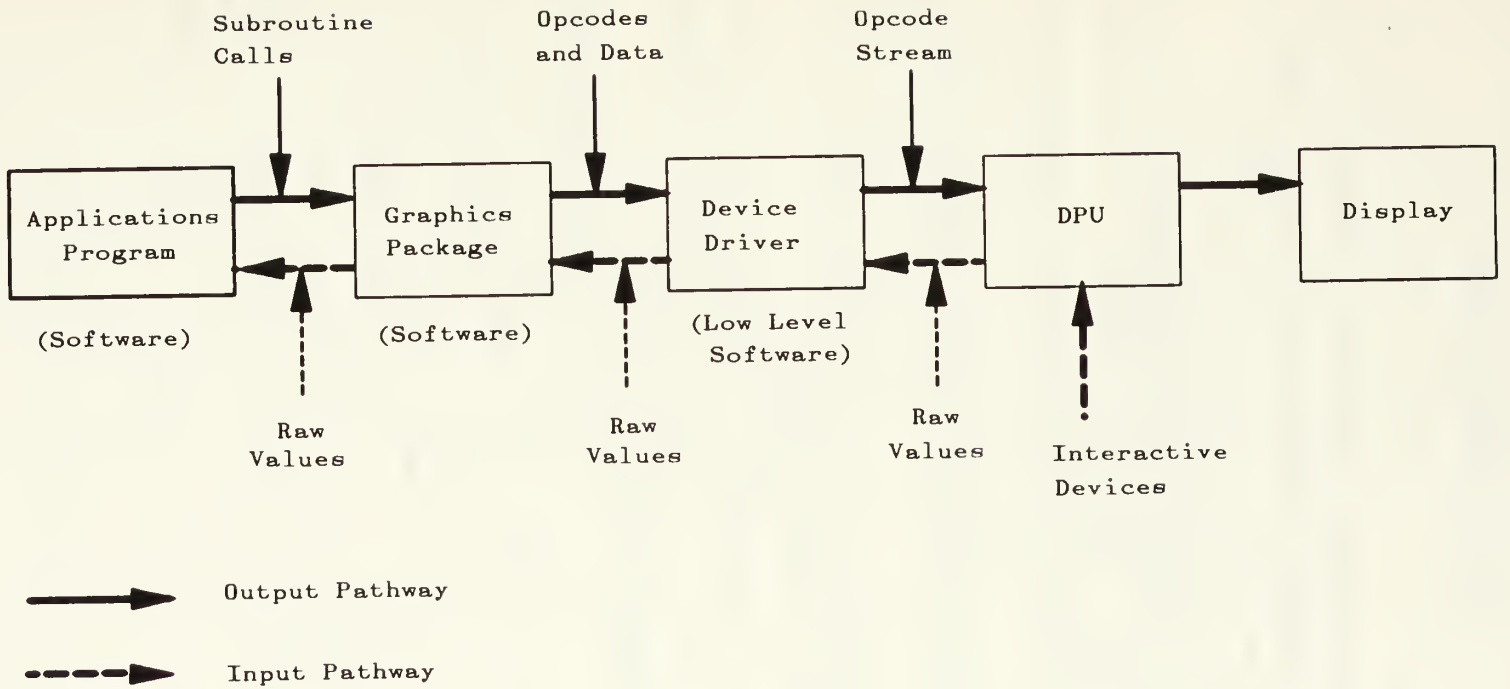


Figure 1.2 Process of Displaying and Manipulating the Picture.

II. RESEARCH RESOURCES

A. COMPUTER SYSTEM

The IRIS Turbo 2400 workstation was chosen as the research tool for this thesis because it is an engineering workstation with powerful general purpose computing and fast response graphics, and is already available at the Naval Postgraduate School's Graphics and Video Laboratory. The system was designed by Silicon Graphics Inc. to combine real-time three-dimensional color graphics with the Unix operating system and Ethernet communication. It incorporates custom VLSI chips in the design to improve speed and reliability. Figure 2.1 shows the IRIS Turbo 2400 system and its significant features.

Conceptually, the system is made up of three pipelined components: the central processing system, the Geometry Pipeline, and the raster subsystem. These three systems communicate over a high-speed private bus. Disk and network communication occurs over the system Multibus. Figure 2.2 shows the block diagram of the IRIS Turbo 2400 architecture.

The central processing unit (CPU) is a Motolora 68020 microprocessor running at 10 MHz. It manages display lists, runs the application program, and controls the Geometry Engine (GE) and raster subsystem.

The Geometry Engine (Figure 2.3) is a custom VLSI processor designed for real-time 3-D graphics. It consists of four 32 bit-floating point ALU's and a microcode control store. The first four engines perform 4-by-4 matrix transformations such as rotation, translation, and scaling. The next four to six engines clip the object in 2 or 3 dimensions. The last two engines perform a perspective division and map the 3-D coordinates to screen space.

The raster subsystem (which consists of three primary components: the frame buffer controller, the update controller, and the display controller) receives the coordinates transformed and clipped by the Geometry Pipeline. It updates the image memory by filling in pixels corresponding to these coordinates, and refreshes the display.

B. IRIS GRAPHICS LIBRARY

The IRIS Graphics Library is a software package that provides high- and low-level support for graphics on the IRIS Workstation. It is the general display processing system that allows an applications programmer to describe and manipulate objects in world coordinate space.

The IRIS Graphics Library consists of graphics and utilities commands. There are eight categories of these commands in the library:

1. Coordinate transformation commands that manipulate a stack of transformation matrices. These commands map a user-defined world coordinate system onto the screen.

2. Primitive drawing commands that draw points, lines, polygons, circles, arcs, curves, and text strings on the screen.

3. Object creation and editing commands that build complicated shapes from simple ones. The newly defined objects can be replicated or modified as desired.

4. Display mode and color map commands that affect the way the IRIS uses its bitplane memory and determine the color used to draw shapes on the screen.

5. Line style, texture, cursor, and font commands that select characteristics for drawing lines, filling polygons, and writing text strings.

6. Input commands that initialize and read input devices that allow the user to poll a device directly or read entries from the event queue.

7. Picking and selecting commands that allow the user to identify the objects that are visible in a given area of the screen.

8. Geometric computing commands that provide access to the computing capabilities of the graphics hardware.

The use of these commands is described in the IRIS User's Guide.

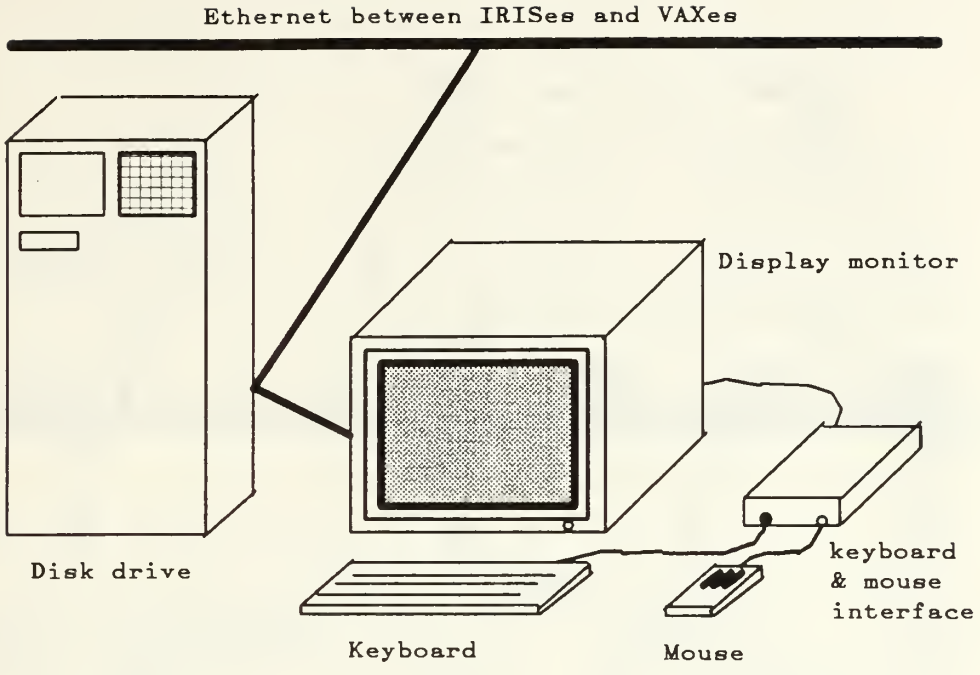
C. MENU PACKAGE

The graphics programs for this study utilize a menu package, designed by Capt. Michael E. Gaddis, USMC. This menu package allows the programmer to specify and use numerous individualized menus for selection of program branches.

The package is very simple to use. It is self contained, in that all procedures are ready to run, except for the initialization procedure. The user describes all

the menus to be used in the program in the initialization procedure. Each a menu has menu number, menu name, number of options, and option names associated with it. The program calls this procedure for menu installation and calls another procedure in the menu package, with specified menu number and screen coordinates (location of the menu) as parameters, for display.

The menu selection mechanism is made through a three-button mouse. The left and right buttons are used for scrolling up and down the menu. The scrolling highlights an option. The middle button activates the highlighted option and causes the program to branch to a procedure specified by that option.



IRIS Turbo 2400 Graphics Workstation:

* 32 bit Motorola 68020 Processor
* 2 MB CPU Memory
* 1024x768x32 bit display memory
* Floating Point Accelerator
* 144 MB Disk Storage
* Cartridge Tape Unit
* Geometry Pipeline with Geometry Egines and Geometry Accelerators
* 60Hz Non-Interlaced Display
* Hardware Smooth Shading
* Unix System V
* IRIS Graphics Library
* Ethernet to VAXes
* 16 bit Z-Buffer for Hidden Surface Elimination
* Digitizer Tablet
* Window Manager with a high-level user interface and overlapping windows for output of both text and interactive 3-D graphics

Figure 2.1 IRIS 2400 System and Its Features.

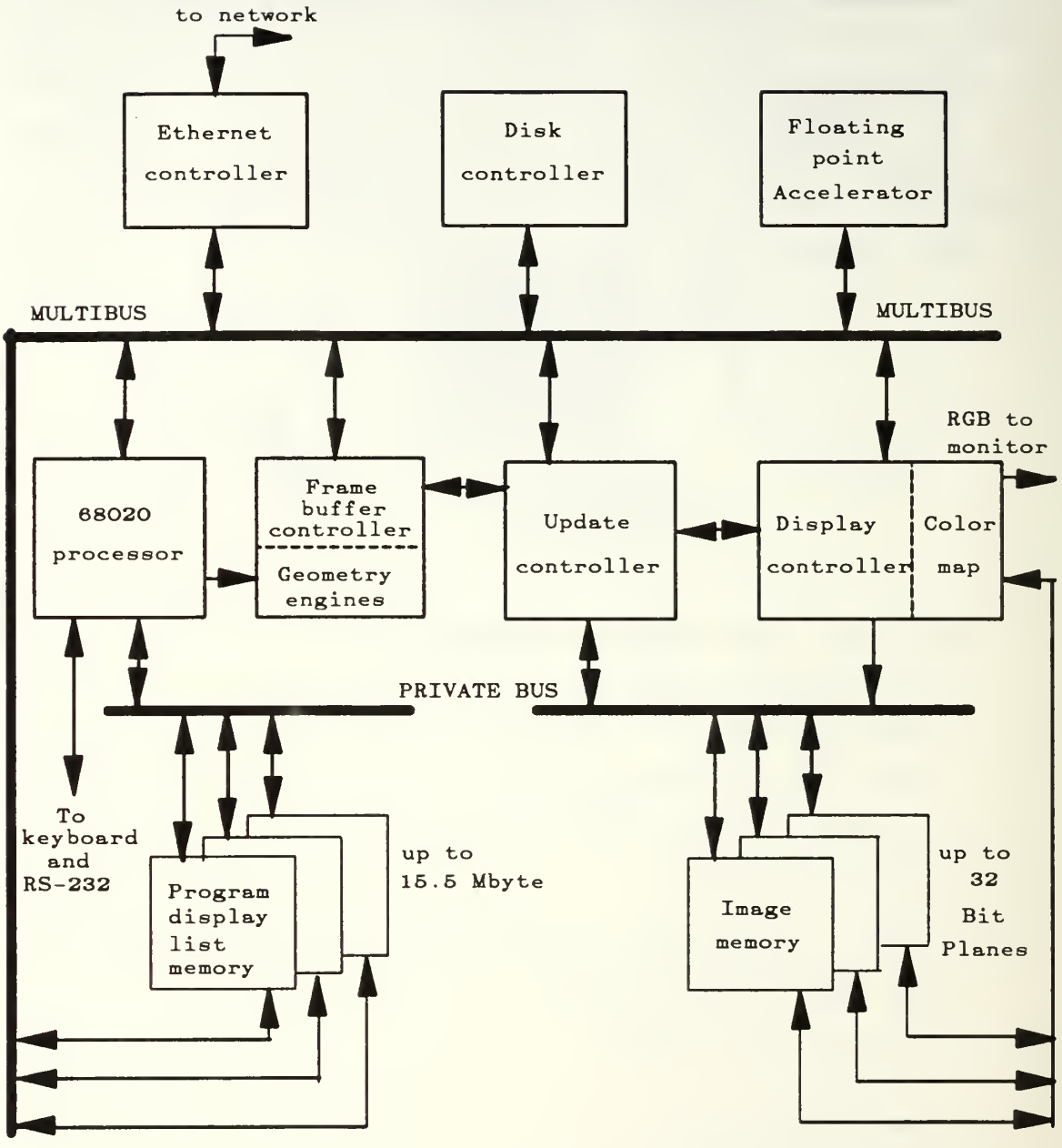


Figure 2.2 Block Diagram of the IRIS Turbo 2400 System.

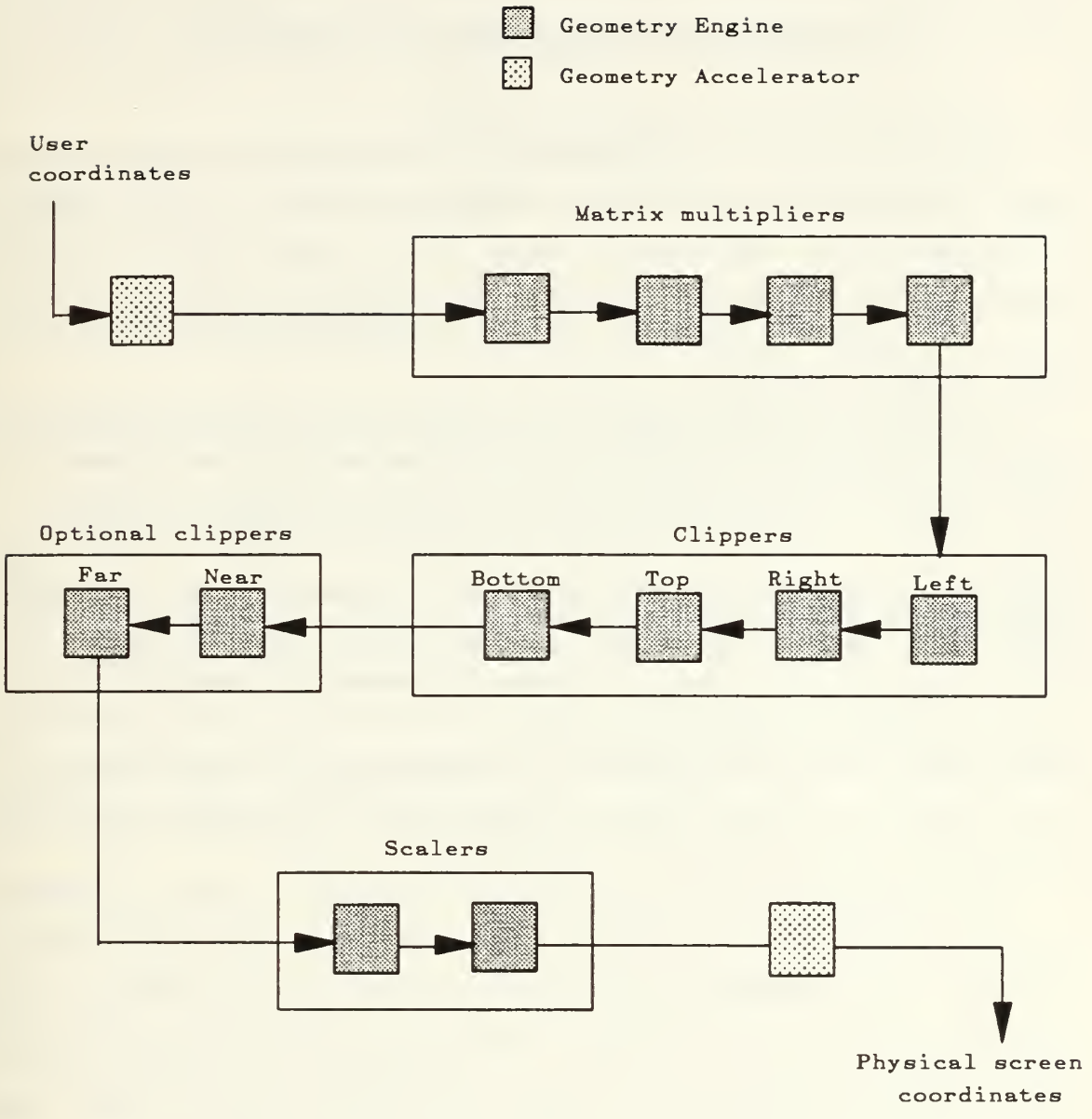


Figure 2.3 Geometry pipeline.

III. BUILDING WALKTHROUGH ALGORITHM

A. BUILDING WALKTHROUGH MECHANISM

The mechanism for building walkthrough is very useful in piping layout design. Such a mechanism allows the designer to look at a particular piping systems within the building model, from different angles and positions, to investigate whether there are any volumetric interferences within the piping system, among other piping system, with the building structures, and with any objects within the building.

In interactive computer graphics, the visual effect of building walkthrough can be simulated by manipulating the picture on the screen through an interactive input device. To simulate building walkthrough the coordinates of a building model are clipped against a specified view volume, perspectively projected onto a projection plane whose normal is parallel to the line of sight defined by a viewpoint and a reference point in the world coordinate space, and finally mapped onto the screen coordinates. The perspective projection is used to provide a degree of realism since it creates an effect similar to that of the human visual system. A mouse is used as an interactive input device to change the coordinates of the viewpoint and the reference point. The change in coordinates of these points causes a change in the picture on the screen.

The walkthrough visual effect produced by interactive computer graphics is analogous to the image produced by a remotely controlled camera that moves in 3-D space. The viewpoint is where the camera is and the line of sight is in the direction that the camera is facing. Turning movement in such a system only requires changes in the line of sight (i.e. the coordinates of the reference point). Moving toward or away from an object, and circling around an object only require changes in the position of the viewpoint and perhaps changes in the position of the reference point. For such movement, the line of sight is maintained. The walkthrough mechanism can be effectively controlled by a three-button mouse device. With three buttons, a mouse can produce up to eight inputs. Seven of these are used for controlling the move-forward, move-backward, turn-left, turn-right, turn-up, turn-down, and circle-around movements. Pressing a button or a combination of buttons produces an input to modify the coordinates of the viewpoint and the line of sight and affects the picture on the screen. The

continuous modification of the viewpoint and the line of sight, that produces the walkthrough visual effect on the screen, is achieved by holding down a button or a combination of buttons. Leaving all three buttons unpressed produces the input that has no effect to the picture on the screen.

B. BUILDING WALKTHROUGH ALGORITHM

The building walkthrough algorithm used in this study is composed of the following outlined steps:

1. **Type of movement determination:** Designate a combination for mouse buttons to each type of walkthrough movement.

2. **Determination of sets of viewpoint-reference point relative positions:** For each type of movement, compare coordinates of the reference point to those of the viewpoint to determine their relative position in the world coordinate system. Define sets of viewpoint-reference point relative positions such that the coordinates of the viewpoint and the reference point can be modified by a set of coordinate modification functions.

3. **Formulation of coordinate modification functions:** Formulate a set of coordinate modification functions used on the coordinates of the viewpoint and the reference point according to the type of movement determined from step 1 and the set of viewpoint-reference point relative positions determined from step 2.

The building walkthrough algorithm developed from the above outlined steps and used in this study is shown in Figures 3.1 - 3.8 .

```
Begin walkthrough algorithm

  create objects
  initialize the viewpoint and the reference point
  specify the display viewport
  specify the perspective viewing pyramid
  specify point and direction of view with the initial viewpoint and reference point
  clear the viewport
  display the objects in the viewport

  while(TRUE)
  begin
    /* test for the type of movement and branch to the
         appropriate movement algorithm to modify the coordinates
         of the viewpoint and the reference point                */

    if all three buttons are hit then
         branch to the circle-around algorithm

    else if the left button and the right button are hit then
         branch to the move-backward algorithm

    else if the middle button and the left button are hit then
         branch to the turn-down algorithm

    else if the middle button and the right button are hit then
         branch to the turn-up algorithm

    else if the left button is hit then
         branch to the turn-left algorithm

    else if the right button is hit then
         branch to the turn-right algorithm

    else if the middle button is hit then
         branch to the move-forward algorithm

    clear the viewport
    specify point and direction of view with new viewpoint and reference point
    display the objects in the viewport

  end

End walkthrough algorithm
```

Figure 3.1 Building Walkthrough Algorithm.

```
Begin circle-around algorithm

  pi=3.1415927   /* constant */
  theta=pi/180.  /* turning angle of 1 degree */

  /* let Vx,Vy,Vz and Rx,Ry,Rz be the coordinates of
     the viewpoint and the reference point respectively.

     d is the distance between the viewpoint and the
     reference point.

     phi2 is the angle between the line of sight and the
     xy-plane.

     phi is the angle between the projected line of sight
     on xz-plane and the x-axis.  */

  /* calculate the values of d, phi2, and phi */
  d = sqrt{ (Vx-Rx)(Vx-Rx) + (Vy-Ry)(Vy-Ry) + (Vz-Rz)(Vz-Rz) }
  phi2 = arcsin{ |Vy-Ry|/d }
  phi  = arcsin{ |Vz-Rz|/(d cos(phi2)) }

  /* first case of viewpoint-reference point relative position */
  if Rx<=Vx and Rz<Vz then

     /* check if the line of sight is still in the case boundary */
     if phi>=theta then
          Vx = Rx + d cos(phi2) cos(phi-theta)
          Vz = Rz + d cos(phi2) sin(phi-theta)
     /* the line of sight crosses the boundary into the second case */
     else
          Vx = Rx + d cos(phi2) cos(theta-phi)
          Vz = Rz - d cos(phi2) sin(theta-phi)

  /* second case of viewpoint-reference point relative position */
  else if Rx<Vx and Rz>=Vz then

     /* check if the line of sight is still in the case boundary */
     if (phi+theta) <= (pi/2.) then
        Vx = Rx + d cos(phi2) cos(phi+theta)
          Vz = Rz - d cos(phi2) sin(phi+theta)
     /* the line of sight crosses the boundary into the third case */
     else
          Vx = Rx - d cos(phi2) sin(theta-pi/2.+phi)
          Vz = Rz - d cos(phi2) cos(theta-pi/2.+phi)
```

Figure 3.2 Circle-around Algorithm.

```
/* third case of viewpoint-reference point relative position */
else if Rx>=Vx and Rz>Vz then

   /* check if the line of sight is still in the case boundary */
   if phi>=theta then
      Vx = Rx - d cos(phi2) cos(phi-theta)
         Vz = Rz - d cos(phi2) sin(phi-theta)
   /* the line of sight crosses the boundary into the fourth case */
   else
         Vx = Rx - d cos(phi2) cos(theta-phi)
         Vz = Rz + d cos(phi2) sin(theta-phi)

/* fourth (last) case of viewpoint-reference point relative position */
else

   /* check if the line of sight is still in the case boundary */
   if (phi+theta) <= (pi/2.) then
         Vx = Rx - d cos(phi2) cos(phi+theta)
         Vz = Rz + d cos(phi2) sin(phi+theta)
   /* the line of sight crosses the boundary into the first case */
   else
         Vx = Rx + d cos(phi2) sin(theta -pi/2.+phi)
         Vz = Rz + d cos(phi2) cos(theta-pi/2.+phi)

End circle-around algorithm
```

Figure 3.2 Circle-around Algorithm (continue).

```
Begin move-backward algorithm

  pi = 3.1415927  /* constant */
  mrate = 0.5 /* moverate of 50 centimeters */

  /* Vx,Vy,Vz are viewpoint coordinates.
     Rx,Ry,Rz are reference point coordinates.
     d is the distance between the viewpoint and the reference point.
     phi2 is the angle between the line of sight and the xz-plane.
     phi is the angle between the projected line of sight on the xz-plane
     and the x-axis.   */

  /* calculate the values of d, phi2, and phi */
  d = sqrt{ (Vx-Rx)(Vx-Rx) + (Vy-Ry)(Vy-Ry) + (Vz-Rz)(Vz-Rz) }
  phi2 = arcsin( |Vy-Ry| /d )
  phi  = arcsin{ |Vz-Rz| /(d cos(phi2)) }

    /* check for the first case of viewpoint-reference point relative
       position and calculate the new x and z coordinates of the
       viewpoint and the reference point. */
    if Rx<=Vx and Rz<Vz then
     Vz = Vz + mrate cos(phi2) sin(phi)
     Vx = Vx + mrate cos(phi2) cos(phi)
     Rz = Rz + mrate cos(phi2) sin(phi)
     Rx = Rx + mrate cos(phi2) cos(phi)

     /* check whether the viewpoint is above or below the reference point
        and calculate the new y coordinates of the viewpoint and the
        reference point  */
     if Ry<Vy then
          Ry = Ry - mrate sin(phi2)
          Vy = Vy - mrate sin(phi2)
     else
          Ry = Ry + mrate sin(phi2)
          Vy = Vy + mrate sin(phi2)

  /* check for the second case of viewpoint-reference point relative
     position and calculate the new x and z coordinates of the
     viewpoint and the reference point. */
  else if Rx<Vx and Rz>=Vz then
     Vz = Vz - mrate cos(phi2) sin(phi)
     Vx = Vx + mrate cos(phi2) cos(phi)
     Rz = Rz - mrate cos(phi2) sin(phi)
     Rx = Rx + mrate cos(phi2) cos(phi)
```

Figure 3.3 Move-backward Algorithm.

```
    /* check whether the viewpoint is above or below the reference point
       and calculate the new y coordinates of the viewpoint and the
       reference point  */
    if Ry<Vy then
       Ry = Ry - mrate sin(phi2)
       Vy = Vy - mrate sin(phi2)
    else
       Ry = Ry + mrate sin(phi2)
       Vy = Vy + mrate sin(phi2)

  /* check for the third case of viewpoint-reference point relative
     position and calculate the new x and z coordinates of the
     viewpoint and the reference point. */
  else if Rx>=Vx and Rz>Vz then
     Vz = Vz - mrate cos(phi2) sin(phi)
     Vx = Vx - mrate cos(phi2) cos(phi)
     Rz = Rz - mrate cos(phi2) sin(phi)
     Rx = Rx - mrate cos(phi2) cos(phi)

     /* check whether the viewpoint is above or below the reference point
        and calculate the new y coordinates of the viewpoint and the
        reference point  */
     if Ry<Vy then
        Ry = Ry - mrate sin(phi2)
        Vy = Vy - mrate sin(phi2)
     else
        Ry = Ry + mrate sin(phi2)
        Vy = Vy + mrate sin(phi2)

  /* check for the fourth case of viewpoint-reference point relative
     position and calculate the new x and z coordinates of the
     viewpoint and the reference point. */
  else
     Vz = Vz + mrate cos(phi2) sin(phi)
     Vx = Vx - mrate cos(phi2) cos(phi)
     Rz = Rz + mrate cos(phi2) sin(phi)
     Rx = Rx - mrate cos(phi2) cos(phi)

     /* check whether the viewpoint is above or below the reference point
        and calculate the new y coordinates of the viewpoint and the
        reference point  */
     if Ry<Vy then
        Ry = Ry - mrate sin(phi2)
        Vy = Vy - mrate sin(phi2)
     else
        Ry = Ry + mrate sin(phi2)
        Vy = Vy + mrate sin(phi2)

End move-backward algorithm
```

Figure 3.3 Move-backward Algorithm (continue).

```
Begin turn-down algorithm

  pi = 3.1415927 /* constant */
  theta = pi/180.  /* turning angle of 1 degree */

  /* Vx,Vy,Vz are the coordinates of the viewpoint.
     Rx,Ry,Rz are the coordinates of the reference point.
     d is the distance between the viewpoint and the reference point.
     phi2 is the angle between the line of sight and the yz-plane.
     phi is the angle between the projected line of sight on the yz-plane
     and the z-axis.  */

  /* calculate the values of d, phi2, and phi */
  d = sqrt{ (Vx-Rx)(Vx-Rx) + (Vy-Ry)(Vy-Ry) + (Vz-Rz)+(Vz-Rz) }
  phi2 = arcsin ( |Vx-Rx| /d )
  phi  = arcsin{ |Vy-Ry| /(d cos(phi2)) }

  /* first case of viewpoint-reference point relative position */
  if Rz<=Vz and Ry>Vy then

     /* check if the line of sight is still in the case boundary */
     if (phi>=theta then
        /* calculate the new y and z coordinates of the reference point */
        Rz = Vz - d cos(phi2) cos(phi-theta)
           Ry = Vy + d cos(phi2) sin(phi-theta)
     /* the line o sight crosses the boundary into the second case */
     else
        /* calculate the new y and z coordinates of the reference point */
           Rz = Vz - d cos(phi2) cos(theta-phi)
           Ry = Vy - d cos(phi2) sin(theta-phi)

  /* second case of viewpoint-reference point relative position */
  else if Rz<Vz and Ry<=Vy then

     /* check if the line of sight is still in the case boundary */
     if (pi/2.-phi) >= theta then
        /* calculate the new y and z coordinates of the reference point */
           Rz = Vz - d cos(phi2) cos(phi+theta)
           Ry = Vy - d cos(phi2) sin(phi+theta)
     /* the line o sight crosses the boundary into the third case */
     else
        /* calculate the new y and z coordinates of the reference point */
           Rz = Vz + d cos(phi2) sin(theta-pi/2.+phi)
           Ry = Vy - d cos(phi2) cos(theta-pi/2.+phi)
```

Figure 3.4 Turn-down Algorithm.

```
/* third case of viewpoint-reference point relative position */
else if Rz>=Vz and Ry<Vy then

  /* check if the line of sight is still in the case boundary */
  if phi >= theta then
    /* calculate the new y and z coordinates of the reference point */
      Rz = Vz + d cos(phi2) cos(phi-theta)
      Ry = Vy - d cos(phi2) sin(phi-theta)
  /* the line o sight crosses the boundary into the fourth case */
  else
    /* calculate the new y and z coordinates of the reference point */
      Rz = Vz + d cos(phi2) cos(theta-phi)
      Ry = Vy + d cos(phi2) sin(theta-phi)

/* fourth (last) case of viewpoint-reference point relative position */
else

  /* check if the line of sight is still in the case boundary */
  if (pi/2.-phi) >= theta then
    /* calculate the new y and z coordinates of the reference point */
      Rz = Vz + d cos(phi2) cos(phi+theta)
      Ry = Vy + d cos(phi2) sin(phi+theta)
  /* the line o sight crosses the boundary into the first case */
  else
    /* calculate the new y and z coordinates of the reference point */
      Rz = Vz - d cos(phi2) sin(theta-pi/2.+phi)
      Ry = Vy + d cos(phi2) cos(theta-pi/2.+phi)

End turn-down algorithm
```

Figure 3.4 Turn-down Algorithm (continue).

```
Begin turn-up algorithm

  pi = 3.1415927   /* constant */
  theta = pi/180.  /* turning angle of 1 degree */

  /* Vx,Vy,Vz are viewpoint coordinates.
     Rx,Ry,Rz are reference point coordinates.
     d is the distance between the viewpoint and the reference point */
     phi2 is the angle between the line of sight and the yz-plane */
     phi is the angle between the projected line of sight on the yz-plane
     and the z-axis.  */

  /* calculate the values of d, phi2, phi */
  d = sqrt{ (Vx-Rx)(Vx-Rx) + (Vy-Ry)(Vy-Ry) + (Vz-Rz)+(Vz-Rz) }
  phi2 = arcsin ( |Vx-Rx| /d )
  phi  = arcsin{ |Vy-Ry| /(d cos(phi2)) }

  /* first case of viewpoint-reference point relative position */
  if Rz<Vz and Ry>=Vy then

     /* check if the line of sight is still in the case boundary */
     if (pi/2.-phi) >= theta then
        /* calculate the new y and z coordinates of the reference point */
           Rz = Vz - d cos(phi2) cos(phi+theta)
           Ry = Vy + d cos(phi2) sin(phi+theta)
     /* the line of sight crosses the boundary into the second case */
     else
        /* calculate the new y and z coordinates of the reference point */
           Rz = Vz + d cos(phi2) sin(theta-pi/2.+phi)
           Ry = Vy + d cos(phi2) cos(theta-pi/2.+phi)

  /* second case of viewpoint-reference point relative position */
  else if Rz>=Vz and Ry>Vy then

     /* check if the line of sight is still in the case boundary */
     if phi >= theta then
        /* calculate the new y and z coordinates of the reference point */
           Rz = Vz + d cos(phi2) cos(phi-theta)
           Ry = Vy + d cos(phi2) sin(phi-theta)
     /* the line of sight crosses the boundary into the third case */
     else
        /* calculate the new y and z coordinates of the reference point */
           Rz = Vz + d cos(phi2) cos(theta-phi)
           Ry = Vy - d cos(phi2) sin(theta-phi)
```

Figure 3.5 Turn-up Algorithm.

```
/* third case of viewpoint-reference point relative position */
else if Rz>Vz and Ry<=Vy then

  /* check if the line of sight is still in the case boundary */
  if (pi/2.-phi) >= theta then
    /* calculate the new y and z coordinates of the reference point */
      Rz = Vz + d cos(phi2) cos(phi+theta)
      Ry = Vy - d cos(phi2) sin(phi+theta)
  /* the line of sight crosses the boundary into the fourth case */
  else
    /* calculate the new y and z coordinates of the reference point */
      Rz = Vz - d cos(phi2) sin(theta-pi/2.+phi)
      Ry = Vy - d cos(phi2) cos(theta-pi/2.+phi)

/* fourth (last)case of viewpoint-reference point relative position */
else

  /* check if the line of sight is still in the case boundary */
  if phi >= theta then
    /* calculate the new y and z coordinates of the reference point */
      Rz = Vz - d cos(phi2) cos(phi-theta)
      Ry = Vy - d cos(phi2) sin(phi-theta)
  /* the line of sight crosses the boundary into the first case */
  else
    /* calculate the new y and z coordinates of the reference point */
      Rz = Vz - d cos(phi2) cos(theta-phi)
      Ry = Vy + d cos(phi2) sin(theta-phi)

End turn-up algorithm
```

Figure 3.5 Turn-up Algorithm (continue).

```
Begin turn-left algorithm
  pi = 3.1415927
  theta = pi/180.  /* turning angle of 1 degree */

  /* Vx,Vy,Vz and Rx,Ry,Rz are coordinates of the viewpoint and
    the reference point respectively.
    d is the distance between the viewpoint and the reference point.
    phi2 is the angle between the line of sight and the xz-plane.
    phi is the angle between the projected line of sight on the xz-plane
    and the x-axis.    */

  /* calculate the values of d, phi2, and phi */
  d = sqrt{ (Vx-Rx)(Vx-Rx) + (Vy-Ry)(Vy-Ry) + (Vz-Rz)(Vz-Rz) }
  phi2 = arcsin( |Vy-Ry|/d )
  phi  = arcsin{ |Vz-Rz|/(d cos(phi2)) }

  /* first case of viewpoint-reference point relative position */
  if Rx<=Vx and Rz<Vz then

    /* check if the line of sight is still in the case boundary */
    if phi >= theta then
      /* calculate the new x znd z coordinates of the viewpoint and the reference point */
        Rx = Vx - d cos(phi2) cos(phi-theta)
        Rz = Vz - d cos(phi2) sin(phi-theta)
    /* the line of sight crosses the boundary into the second case */
    else
      /* calculate the new x znd z coordinates of the viewpoint and the reference point */
        Rx = Vx - d cos(phi2) cos(theta-phi)
        Rz = Vz + d cos(phi2) sin(theta-phi)

  /* second case of viewpoint-reference point relative position */
  else if Rx<Vx and Rz>=Vz then

    /* check if the line of sight is still in the case boundary */
    if (pi/2.-phi) >= theta then
      /* calculate the new x znd z coordinates of the viewpoint and the reference point */
        Rz = Vz + d cos(phi2) sin(phi+theta)
        Rx = Vx - d cos(phi2) cos(phi+theta)
    /* the line of sight crosses the boundary into the third case */
    else
      /* calculate the new x znd z coordinates of the viewpoint and the reference point */
        Rx = Vx + d cos(phi2) sin(theta-pi/2.+phi)
        Rz = Vz + d cos(phi2) cos(theta+pi/2.+phi)
```

Figure 3.6 Turn-left Algorithm.

```
/* third case of viewpoint-reference point relative position */
else if Rx>=Vx and Rz>Vz then

  /* check if the line of sight is still in the case boundary */
  if phi >= theta then
    /* calculate the new x znd z coordinates of the viewpoint and the reference point */
      Rx = Vx + d cos(phi2) cos(phi-theta)
      Rz = Vz + d cos(phi2) sin(phi-theta)
  /* the line of sight crosses the boundary into the fourth case */
  else
    /* calculate the new x znd z coordinates of the viewpoint and the reference point */
      Rx = Vx + d cos(phi2) cos(theta-phi)
      Rz = Vz - d cos(phi2) sin(theta-phi)

/* fourth (last)  case of viewpoint-reference point relative position */
else

  /* check if the line of sight is still in the case boundary */
  if (pi/2.-phi) >= theta then
    /* calculate the new x znd z coordinates of the viewpoint and the reference point */
      Rx = Vx + d cos(phi2) cos(phi+theta)
      Rz = Vz - d cos(phi2) sin(phi+theta)
  /* the line of sight crosses the boundary into the first case */
  else
    /* calculate the new x znd z coordinates of the viewpoint and the reference point */
      Rx = Vx - d cos(phi2) sin(theta-pi/2.+phi)
      Rz = Vz - d cos(phi2) cos(theta-pi/2.+phi)

End turn-left algorithm
```

Figure 3.6 Turn-left Algorithm (continue).

```
Begin turn-right algorithm
  pi = 3.1415927
  theta = pi/180.  /* turning angle of 1 degree */

  /* Vx,Vy,Vz and Rx,Ry,Rz are coordinates of the viewpoint and
     the reference point respectively.
     d is the distance between the viewpoint and the reference point.
     phi2 is the angle between the line of sight and the xz-plane.
     phi is the angle between the projected line of sight on the xz-plane
     and the x-axis.  */

  /* calculate the values of d, phi2, and phi */
  d = sqrt{ (Vx-Rx)(Vx-Rx) + (Vy-Ry)(Vy-Ry) + (Vz-Rz)(Vz-Rz) }
  phi2 = arcsin( |Vy-Ry| /d )
  phi  = arcsin{ |Vz-Rz| /(d cos(phi2)) }

  /* first case of viewpoint-reference point relative position */
  if Rx<Vx and Rz<=Vz then

    /* check if the line of sight is still in the case boundary */
    if (pi/2.-phi) >= theta then

      /* calculate the new coordinates of the reference point */
        Rx = Vx - d cos(phi2) cos(phi+theta)
        Rz = Vz - d cos(phi2) sin(phi+theta)
    /* the line of sight crosses the boundary into the second case */
    else
      /* calculate the new coordinates of the reference point */
        Rx = Vx + d cos(phi2) sin(theta-pi/2.+phi)
        Rz = Vz - d cos(phi2) cos(theta-pi/2.+phi)

  /* second case of viewpoint-reference point relative position */
  else if Rx<=Vx and Rz>Vz then

     /* calculate the new coordinates of the reference point */
    if phi >= theta then
      /* calculate the new coordinates of the reference point */
        Rz = Vz - d cos(phi2) cos(phi-theta)
        Rx = Vx + d cos(phi2) sin(phi-theta)
    /* the line of sight crosses the boundary into the third case */
    else
      /* calculate the new coordinates of the reference point */
        Rx = Vx - d cos(phi2) cos(theta-phi)
        Rz = Vz - d cos(phi2) sin(theta-phi)
```

Figure 3.7 Turn-right Algorithm.

```
/* third case of viewpoint-reference point relative position */
else if Rx>Vx and Rz>=Vz then

   /* calculate the new coordinates of the reference point */
  if (pi/2.-phi) >= theta then
    /* calculate the new coordinates of the reference point */
      Rx = Vx + d cos(phi2) cos(phi+theta)
      Rz = Vz + d cos(phi2) sin(phi+theta)
  /* the line of sight crosses the boundary into the fourth case */
  else
    /* calculate the new coordinates of the reference point */
      Rx = Vx - d cos(phi2) sin(theta-pi/2.+phi)
      Rz = Vz + d cos(phi2) cos(theta-pi/2.+phi)

/* fourth (last) case of viewpoint-reference point relative position */
else

   /* calculate the new coordinates of the reference point */
  if phi >= theta then
    /* calculate the new coordinates of the reference point */
      Rx = Vx + d cos(phi2) cos(phi-theta)
      Rz = Vz - d cos(phi2) sin(phi-theta)
  /* the line of sight crosses the boundary into the first case */
  else
    /* calculate the new coordinates of the reference point */
      Rx = Vx + d cos(phi2) cos(theta-phi)
      Rz = Vz + d cos(phi2) sin(theta-phi)

End turn-right algorithm
```

Figure 3.7 Turn-right Algorithm (continue).

```
Begin move-forward algorithm

  pi = 3.1415927  /* constant */
  mrate = 0.5 /* move rate of 50 centimeters */

  /* Vx,Vy,Vz are view point coordinates.
     Rx,Ry,Rz are reference point coordinates.
     d is the distance between the viewpoint and the reference point.
     phi2 is the angle between the line of sight and the xz-plane.
     phi is the angle between the projected line of sight on the xz-plane
     and the x-axis.  */

  /* calculate the values of d, phi2, and phi */
  d = sqrt{ (Vx-Rx)(Vx-Rx) + (Vy-Ry)(Vy-Ry) + (Vz-Rz)(Vz-Rz) }
  phi2 = arcsin( |Vy-Ry| /d )
  phi  = arcsin{ |Vz-Rz| /(d cos(phi2)) }

  /* check for the first case of viewpoint-reference point relative position
     and calculate the new x and z coordinates of the viewpoint and
     the reference point.  */
  if Rx<=Vx and Rz<Vz then
     Vz = Vz - mrate cos(phi2) sin(phi)   /* Vx,Vy,Vz are viewpoint coord */
     Vx = Vx - mrate cos(phi2) cos(phi)
     Rz = Rz - mrate cos(phi2) sin(phi) /*Rx,Ry,Rz are reference point coord*/
     Rx = Rx - mrate cos(phi2) cos(phi)

     /* check whether the viewpoint is above or below the reference point
        and calculate the new y coordinates of the viewpoint and the
        reference point.  */
     if Ry<Vy then
          Ry = Ry - mrate sin(phi2)
          Vy = Vy - mrate sin(phi2)
     else
          Ry = Ry + mrate sin(phi2)
          Vy = Vy + mrate sin(phi2)

  /* check for the second case of viewpoint-reference point relative position
     and calculate the new x and z coordinates of the viewpoint and
     the reference point.  */
  else if Rx<Vx and Rz>=Vz then
     Vz = Vz + mrate cos(phi2) sin(phi)
     Vx = Vx - mrate cos(phi2) cos(phi)
     Rz = Rz + mrate cos(phi2) sin(phi)
     Rx = Rx - mrate cos(phi2) cos(phi)
```

Figure 3.8 Move-forward Algorithm.

```
    /* check whether the viewpoint is above or below the reference point
       and calculate the new y coordinates of the viewpoint and the reference point */
    if Ry<Vy then
       Ry = Ry - mrate sin(phi2)
          Vy = Vy - mrate sin(phi2)
    else
          Ry = Ry + mrate sin(phi2)
          Vy = Vy + mrate sin(phi2)

  /* check for the third case of viewpoint-reference point relative position
     and calculate the new x and z coordinates of the viewpoint and the reference point */
  else if Rx>=Vx and Rz>Vz then
     Vz = Vz + mrate cos(phi2) sin(phi)
     Vx = Vx + mrate cos(phi2) cos(phi)
     Rz = Rz + mrate cos(phi2) sin(phi)
     Rx = Rx + mrate cos(phi2) cos(phi)

     /* check whether the viewpoint is above or below the reference point
        and calculate the new y coordinates of the viewpoint and the reference point */
     if Ry<Vy then
          Ry = Ry - mrate sin(phi2)
          Vy = Vy - mrate sin(phi2)
     else
          Ry = Ry + mrate sin(phi2)
          Vy = Vy + mrate sin(phi2)

  /* check for the fourth case of viewpoint-reference point relative position
     and calculate the new x and z coordinates of the viewpoint and the reference point */
  else
     Vz = Vz - mrate cos(phi2) sin(phi)
     Vx = Vx + mrate cos(phi2) cos(phi)
     Rz = Rz - mrate cos(phi2) sin(phi)
     Rx = Rx + mrate cos(phi2) cos(phi)

     /* check whether the viewpoint is above or below the reference point
        and calculate the new y coordinates of the viewpoint and the reference point */
     if Ry<Vy then
          Ry = Ry - mrate sin(phi2)
          Vy = Vy - mrate sin(phi2)
     else
          Ry = Ry + mrate sin(phi2)
          Vy = Vy + mrate sin(phi2)

End move-forward algorithm
```

Figure 3.8 Move-forward Algorithm (continue).

IV. PIPING LAYOUT ALGORITHM

A. THREE DIMENSIONAL PIPING LAYOUT TECHNIQUE

Three-dimensional piping layout can be accomplished via an interactive computer graphics system. The method for such an operation involves mapping an image of the current piping system into a viewport on a graphics screen, overlaying an image of a piping component in the same viewport, manipulating the overlay image of the piping component with an interactive input device, and updating the piping system.

There is one major difficulty associated with 3-D piping layout via an interactive computer graphics system. This difficulty is with respect to the problem of piping positioning which requires the translation and rotation of the image of a 3-D piping component to a desired position and orientation on the 2-D graphics screen. A poor graphical manipulation technique can result in a hardly manageable and/or poorly performing system.

This study presents one 3-D piping layout technique. The technique is menu driven, with each selection being made through a three-button mouse device. Our piping positioning method uses a technique which dynamically moves an object around with a mouse device. This technique is known as dragging. The transformation (translation and rotation) of the overlay image of a piping component is controlled by valuators provided by the mouse. Though a mouse can provide only two valuators (horizontal and vertical), a menu can be used to support a transformation selection and only one valuator is sufficient for controlling a transformation.

Three views of the object space are used to eliminate ambiguity. The orthogonal projections of the top view, front view, and side view of the layout model can be selected for display during piping positioning manipulation to clarify the position and orientation of each piping component in the piping system. An axis, in the screen coordinate system, is also used to provide directions for translations and rotations of a piping component.

B. THREE DIMENSIONAL PIPING LAYOUT ALGORITHM

The piping layout algorithm is composed of the following outlined steps:

1. **Piping system selection:** Design and install a menu for piping system selection. Each option identifies a piping system to be laid out or modified.

2. **Piping component type selection:** Design and install a menu for piping component type selection. Each option identifies a type of a piping component. The standard types used are straight pipe, elbow, tee, and reducer.

3. **Dimension selection:** Design and install menus for pipe diameter selection and elbow turning angle selection. Drawing a straight pipe requires the diameter and the length of the pipe. The length is interactively defined by the distance between two points on the graphics screen marked by the cursor. Drawing an elbow component requires a diameter and a turning angle. Drawing a tee component and a reducer component require a set of two diameters.

4. **Display of the piping system and piping components:** Display the image of the existing piping system according to the system identification selected from Step 1. Then display the piping component according to the type and dimensions determined in Step 2 and Step 3.

5. **Position and orientation adjustment:** Design and install a menu for a piping component position and orientation adjustment. The menu includes translation in x-direction, y-direction, z-direction, and rotations about x-axis, y-axis, z-axis, insertion, and deletion options. Each option with the exception of the insertion and deletion options, when selected, performs a type of transformation. The rate of transformation is determined by the position of a valuator provided by a mouse device. Three views (top, front, side) of the layout model are displayed to eliminate ambiguity during the position and orientation adjustment process.

6. **Insertion and deletion of a piping component:** After a piping component is transformed to the desired position, it can be saved and added to the existing piping layout by selecting the insertion option of the menu in Step 5. If the piping component is incorrect, it can be deleted from the screen at any time by selecting the deletion option of the same menu.

The three dimensional piping layout algorithm developed from the above outlined steps and used in this study is in Figures 4.1 - 4.6 .

```
Begin Piping Algorithm

  Display the pipe system selection menu

  select an option from the menu

  if the EXIT option is selected then

    clear the screen and branch out of the algorithm to the IRIS system

  else if the SPRINKLER option is selected then
    system = 1
    branch to the View the Model Algorithm

  else if the WATER SUPPLY/DRAINAGE option is selected then
    system = 2
    branch to the View the Model Algorithm

  else if COMPLETE PIPE SYSTEM is selected then
    system = 3
    branch to the View the Model Algorithm

  else if the NEW SYSTEM is selected then
    system = 4
    branch to the View the Model Algorithm

End Piping Algorithm
```

Figure 4.1 Piping Algorithm.

```
Begin View the Model Algorithm
  determine the piping system selected by system value from the Piping algorithm
  initialize the scale factor, translation amount, and rotation amount
  display the image of the current piping system in the display viewport
  display the "view the model" menu in the menu viewport

  while(TRUE)
  begin

    select an option from the menu

    if the TOP VIEW option is selected then
         rotate the modified image (if any) of the current piping system 90 degrees
         about x-axis and display it in the display viewport
    else if the SIDE VIEW option is selected then
         rotate the modified image (if any) of the current piping system -90 degrees
         about y-axis and display it in the display viewport
    else if the FRONT VIEW option is selected then
         display the modified (if any ) image of the current piping system
         in the display viewport
    else if the PERSPECTIVE VIEW option is selected then
         rotate the modified (if any) image of the current piping system
         30 degrees about x-axis and 30 degrees about y-axis and display
         it in the display viewport
    else if the SCALE option is selected then
         calculate the scaling factor from the position of the mouse valuator,
         display the modified image of the current piping system in the
         display viewport
    else if the MOVE AROUND option is selected the
         calculate the translation amounts in x,y  from mouse valuators,
         display the modified image of the current piping system
    else if the ROTATE option is selected then
         calculate the amount of rotation from the position of a mouse valuator,
         display the modified image of the current piping system
    else if the BFRAME option is selected then
         display the modified (if any) image of the current piping system
         if the building structural frame is not displayed then display it
         else if it is already displayed then remove it from the screen
    else if the DRAW PIPE option is selected then
         reinitialize the scale factor, the amount of translation, the amount of
         rotation, and display the initial image of the current piping system
         and branch out of the algorithm to the Drawpipe Algorithm
    else if the WALKTHROUGH option is selected then
         branch out of the algorithm to the Walkthrough Algorithm
    else if the EXIT option is selected then
         branch out of the algorithm to the Piping System Algorithm
  end

End View the Model Algorithm
```

Figure 4.2 View the Model Algorithm.

```
Begin Drawpipe Algorithm

  display the pipe type menu in the menu viewport

  select an option from the menu

    if the STRAIGHT PIPE option is selected then
         type = 1
         branch out of the algorithm to the Dimension Algorithm

    if the ELBOW option is selected then
         type = 2
         branch out of the algorithm to the Dimension Algorithm

    if the TEE option is selected then
         type = 3
         branch out of the algorithm to the Dimension Algorithm
    if the REDUCER option is selected then
         type = 4
         branch out of the algorithm to the Dimension Algorithm

    if the DELETE option is selected then
         delete the last drawn piping component from the current piping system,
         recursively branch back to the beginning of the algorithm

    if the VIEW option is selected then
         branch out of the algorithm to the View the Model Algorithm

    if the EXIT option is selected then
         branch out of the algorithm to the Piping System Algorithm


End Drawpipe Algorithm
```

Figure 4.3 Drawpipe Algorithm.

```
Begin Dimension Algorithm

  check the piping component type value passed from the drawpipe algorithm

  if type = 1 or type = 2 then
    display the pipe diameter selection menu

    select an option from the menu

        if a pipe diameter is selected then
          mark a point on the screen with the cursor and a mouse button to
          indicate the initial position of the drawing piping component
          if type = 1 then
            mark another point to determine the length of the pipe,
            calculate the length, and branch out of the algorithm and pass
            the diameter value and the length value to the Insertpipe Algorithm

        else if type = 2 then
            display the turn angle selection menu
            select an option for the turn angle value, branch out of the
            algorithm and pass the selected turn angle value to the Insertpipe Algorithm

      else if the VIEW option is selected then
            branch out of the algorithm to the View the Model Algorithm


  else if type = 3 or type = 4 then
    display the pipe diameters selection menu

    select an option from the menu

        if a pair of diameters is selected then
          branch out of the algorithm and pass the diameter values to
          the Insertpipe Algorithm

      else if the EXIT option is selected then
            branch out of the algorithm to the Piping System Algorithm


End Dimension Algorithm
```

Figure 4.4 Dimension Algorithm.

```
Begin Insertpipe Algorithm

  display three views of the current piping system and three views of the
  overlay image of the selected piping component with x-,y-,z- axes in the
  display viewport

  display the position adjustment menu in the menu viewport

  while(TRUE)
  begin

    select an option from the menu

    if a MOVE option (MOVE in x-, y-, or z- direction) is selected then
         calculate the translation amount from the position of a mouse valuator,
         display three views of the current piping system  and the modified
         three views of the overlay image of the piping component and axes

    else if a ROTATE option (ROTATE about x-, y-, or z- axis) is selected then
         calculate the rotation amount from the position of a mouse valuator,
         display three views of the current piping system and the modified
         three views of the overlay image of the piping component and axes

    else if the VIEW SELECTION option is selected then
         branch out of the algorithm to the Selectview Algorithm

    else if the SAVE option is selected then
         insert the piping component into the current piping system,
         display the updated current piping system without axes, and
         branch out of the algorithm to the Drawpipe Algorithm

    else if the DELETE option is selected then
         delete the overlay image of the piping component and axes from the screen

    else if the EXIT option is selected then
         branch out of the algorithm to the Piping System Algorithm

  end

End Insertpipe Algorithm
```

Figure 4.5 Insertpipe Algorithm.

```
Begin Selectview Algorithm

  display the view selection menu in the menu viewport

  while(TRUE)
  begin

    select an option from the menu

    if the TOP VIEW option is selected then
      display the top view of the piping system and the top view of the
      overlay image of the piping component in the display viewport

    else if the FRONT VIEW option is selected then
      display the front view of the piping system and the front view of the
      overlay image of the piping component in the display viewport

    else if the SIDE VIEW option is selected then
      display the side view of the piping system and the side view of the
      overlay image of the piping component in the display viewport

    else if the THREE VIEWS option is selected then
      display the three views of the piping system and the three views of the
      overlay image of the piping component in the display viewport

    else if the EXIT option is selected then
        branch out of the algorithm to the Insertpipe Algorithm

  end

End Selectview Algorithm
```

Figure 4.6 Selectview Algorithm.

V. PROGRAM IMPLEMENTATION DETAILS

A multi-file program, the implementation of the Walkthrough Algorithm and the 3-D Piping Layout Algorithm in the C programming language, is included in the appendix of this study. The program runs on the IRIS Turbo 2400 Workstation under the command "building".

The execution of the command "building" displays the program title (Figure 5.1) on the screen. From this point, the rest of the operation is self evident from the display menus. The mouse is the only interactive input device. Hitting a mouse button changes the picture on the screen to the picture of the Piping System Menu (Figure 5.2).

The Piping System Menu is designed to offer five options: Sprinkler System, Water Supply/Drainage System, Complete Piping System, New Piping System, and Exit to the System. The Sprinkler system, the Water Supply/Drainage System, and the Complete Piping System are predefined for the test runs. The New Piping System is the layout model without an existing piping system. This option is used for the demonstration of a piping layout operation, via an interactive computer graphics system. The Exit to the System option terminates the program.

The user begins the piping layout operation by selecting a piping system option from the Piping System Menu. The selection of a piping system clears the screen then displays the View the Model Menu (Figure 5.3) in the menu viewport, and displays a layout model with the selected piping system in the display viewport.

The View the Model Menu presents the user with options to view the top view, front view, side view, or perspective view of the layout model, options to scale the image of the layout model up or down, to move the scaled image around on the screen, and to rotate it about the y-axis. Other available options are the Drawpipe option, the Walkthrough option, and the Exit to Piping System options.

The selection of the Drawpipe option, from the View the Model Menu, displays the Piping Subsystem Menu (Figure 5.4). This menu is used to specify the color of a piping component. In this study, cold water pipes are drawn in blue, hot water pipes are drawn in red, and drainage pipes are drawn in black. After the selection of an option to indicate the color of a piping component, the

Component Type Menu (Figure 5.5) is displayed for the selection of the piping component to be drawn. The Component Type Menu offers four types of components: straight pipe, elbow, tee, and reducer. Each type requires some dimensions before it can be drawn. The straight pipe requires a pipe diameter which is selected from a menu, and the length of the pipe which is specified by marking two points on the screen as directed by the program. The elbow component requires a pipe diameter and a turning angle, both are selected from menus prompted by the program. The tee component and the reducer component require a set of two diameters, selected from a menu. Before a piping component is drawn, the program instructs the user to mark the position of the component to be drawn on the screen. A grid is overlaid in the display viewport to aid the user in specifying the piping position (xy-plane). An image of a piping component is overlaid in the display viewport after piping dimensions and position are specified.

The position and orientation of the overlay image of the piping component is adjusted via the Position Adjustment Menu (Figure 5.6). The menu offers options for viewing the overlay image of the piping component in a particular view (top, front, side), for moving and rotating the overlay image of the piping component to a desired position and orientation. The overlay image of the piping component is added to the current piping system by the selection of the Insert option, or removed from the screen by the selection of the Delete option from this menu.

After the insertion or deletion of a piping component, the View the Model Menu is displayed again so that the modified piping system can be viewed from different viewpoints. At this point, the previously inserted pipes can be deleted, in the reverse order of their insertions, starting from the most recently inserted pipes, by the selection of the View the Model option. The user enters the building walkthrough mode to view the modified piping system in a walkthrough fashion by the selection of the Walkthrough option.

The selection of the Walkthrough option clears the screen and sets it up for the walkthrough operation. The set up divides the screen into four display areas (Figure 5.7). The first area, the largest one, is used to display the layout model (a building model with a selected piping system). The visual effect for a building walkthrough movement, occurs in this display area. The second area is used to display static images of the orthogonal projections of the front view and the side view of the layout model. The third display area is used to display a perspective projection of the layout model. An image of a marker indicates the relative

position of the viewer to objects in the world coordinate system. The fourth area is used to display an instruction for building walkthrough control, an instruction for picking an option, and pick options. The cursor is initially displayed in the first area at the bottom of the screen and can be moved around with the mouse.

After the screen is set up for the building walkthrough operation, the mouse becomes the main control. Its buttons are used in generating inputs for walkthrough movements. Its movement moves the cursor on the screen. When the cursor is moved inside a pick option boundary box and the left mouse button is pressed, the selected option is made. When it is outside pick option boundary boxes, holding down a mouse button or a combination of mouse buttons causes changes in the picture on the screen. This action produces the visual effect of the selected building walkthrough movement. The walkthrough control instruction is displayed in the fourth display area. It provides information for generating an input for each type of building walkthrough.

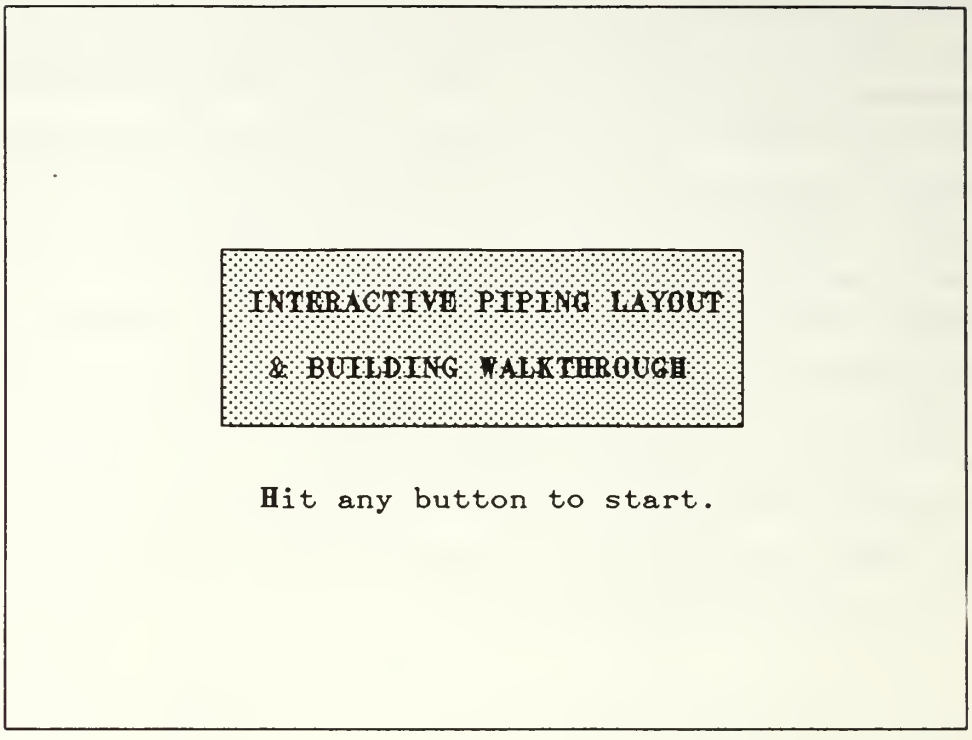


Figure 5.1 The Tile of the Program.

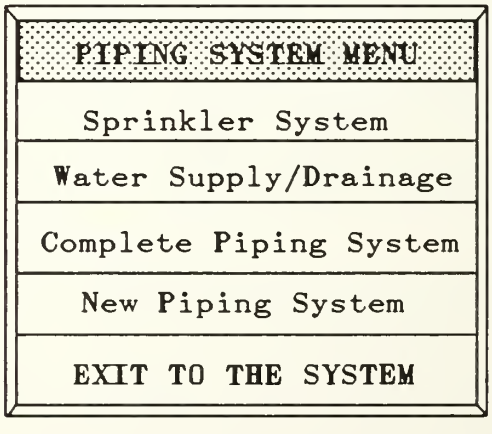


Figure 5.2 Piping System Menu.

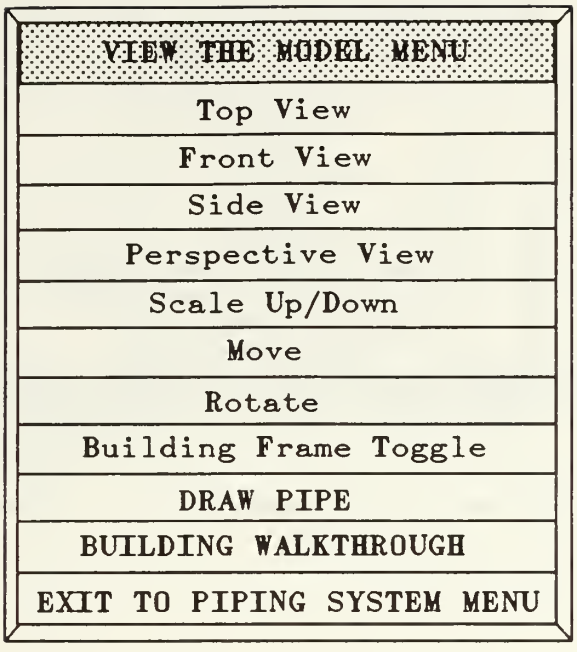


Figure 5.3 View the Model Menu.

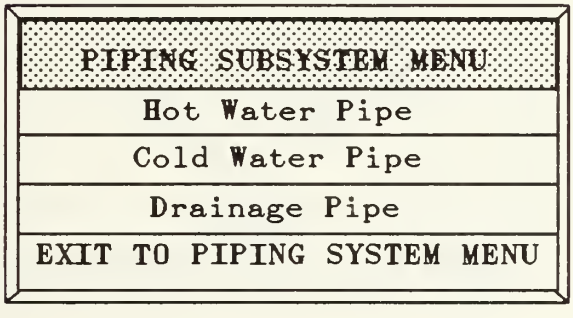


Figure 5.4 Piping Subsystem Menu.

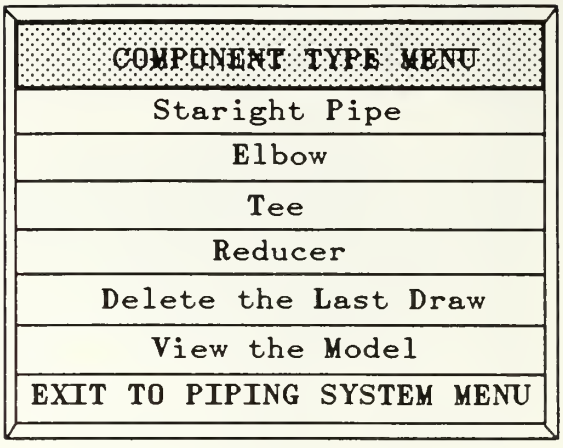


Figure 5.5 Component Type Menu.

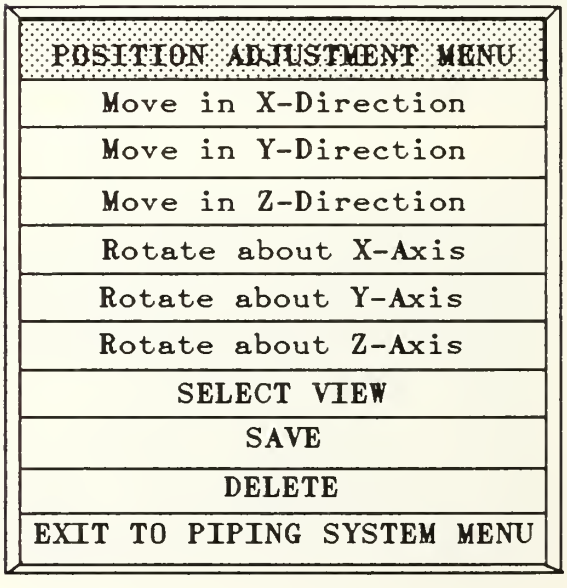


Figure 5.6 Position Adjustment Menu.

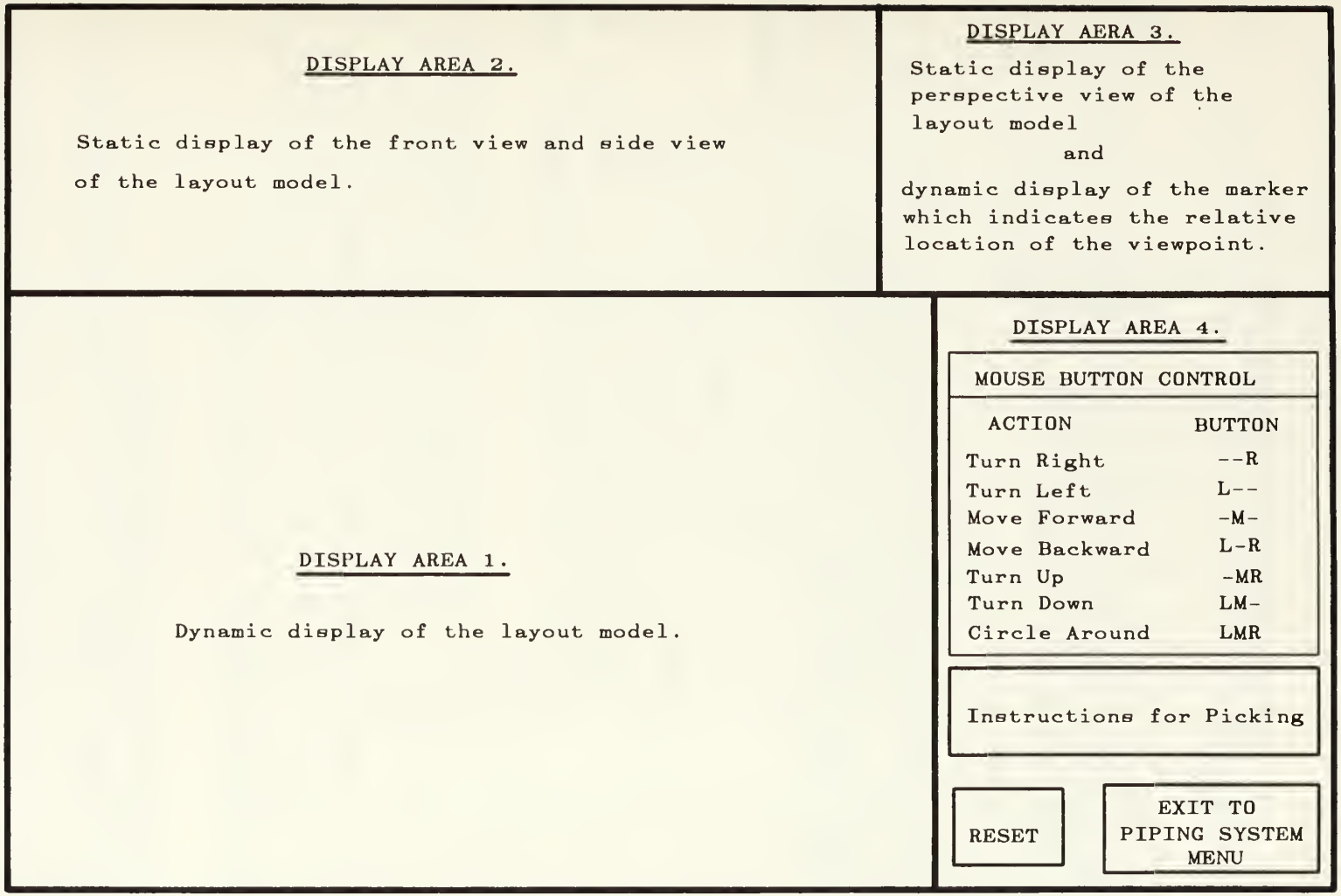


Figure 5.7 Building Walkthrough Screen Set up.

VI. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

This study demonstrates the interactive techniques for a building walkthrough mechanism and for the placement of 3-D piping into a 3-D building using a 2-D graphics display and a mouse device. The test run results indicate that the performance (speed) of building walkthrough and piping positioning mechanisms depends on the model complexity. The model which includes the entire piping system requires much more time to produce a change in the picture than one that includes just a particular piping system. It is recommended that the designer select a single piping system for manipulation for fast interactive response. Each layout model with a particular piping system can be viewed together either in building walkthrough fashion or by manipulating the image of the layout model on the screen. Though the test run results indicate that the performance of the walkthrough mechanism depends on the complexity of the layout model, the techniques are, in some extent, useful and viable to a piping layout task. The technique of placing 3-D piping into a 3-D building is made simple and efficient, by using a menu package to generate input commands. The use of the mouse valuator to control the image transformation for piping positioning is fairly accurate.

The study also illustrates the capabilities of interactive computer graphics as a computer aided design tool in a piping layout application. The drawing and modification of a piping layout system is accomplished much faster than conventional drafting techniques. Unlike conventional drafting techniques which limit one to only three views (top, front, side), the layout model drawn by an interactive computer graphics system can be viewed from many different viewpoints in the world space and at any desired scale factor.

B. LIMITATIONS AND RECOMMENDATIONS

The Walkthrough Algorithm and the 3-D Piping Algorithm developed and used in this study have some limitations:

1. The algorithms do not support hidden surface elimination. The components of the layout models are wire frame drawings and are displayed in double buffer mode for fast response interaction. The images of objects on the screen depend on the order of the display commands. The image of an object, in spite of the closer distance to the viewer, can not be seen if it is drawn before the larger image of another object that is drawn at the same

coordinate location. Even though three views of the object space are displayed, ambiguity still exists in a complex piping system.

2. The algorithms do not utilize a file system to support the updating of the layout model. Any modification to a piping system is maintained during the walkthrough and the 3-D piping layout operation, as long as the program does not exit to the UNIX system.

3. The algorithms do not utilize a pick mechanism for the identification of objects in the world coordinate system that require editing. Hence the deletion of a piping component from a piping system is designed to default to the most recently inserted component. However, deletion of a number of inserted components can be done one at a time according to the sequence of their insertion.

With these limitations, the interactive techniques and algorithms presented in this study are not yet practical for a layout task in a complex industrial processing plant. Additional studies in the areas of hidden surface elimination, file access, and graphical editing are recommended for the improvement of the algorithms.

BIBLIOGRAPHY

Foley J. D., and Van Dam A., Fundamentals of Interactive Computer Graphics , Addison-Wesley, 1984.

Gardan Y., Lucas M., and Budynas R. G., Interactive Graphics in CAD , Unipub, 1984.

Naval Postgraduate School Report NPS 52-85-012, Workstation Graphics Capabilities for the 1990's and Beyond , by M. J. Zyda, September 1985.

Silicon Graphics Incorporation, IRIS User's Guide Version 2.0 Update , document number 5001-051-001-1, 1985.

Gaddis M. E., Menu Package ,program written for the IRIS 2400 System at the Naval Postgraduate School, 1985.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Chairman (Code 52) Department of Computer Science Naval Postgraduate School Monterey, California 93943	1
4.	Computer Technology Programs (Code 37) Naval Postgraduate School Monterey, California 93943	1
5.	Michael J. Zyda (Code 52Zk) Department of Computer Science Naval Postgraduate School Monterey, California 93943	2
6.	LCDR Paul W. Callahan (Code 52Cs) Department of Computer Science Naval Postgraduate School Monterey, California 93943	1
7.	Office of the Air Attache' The Royal Thai Embassy 5600 16th St., N.W. Washington D.C. 20011	2
8.	FLT LT Surasak Mungsing 13/2 Tanintorn Village Wipawadee-Rangsit Rd., Bangkok 10210, Thailand	2

217522

Thesis
M927 Mungsing
c.1 A software implementation of an interactive graphics system for three dimension modeling and layout.